# Sprint 1: Learning outcomes journal

James Browne

Contents

[Sprint 1: Learning outcomes journal 1](#_Toc340101841)

[LO: Creating a skybox in Three.js 2](#_Toc340101842)

[LO: Adding models to the scene 3](#_Toc340101843)

[LO: Collision detection using Three.js bounding spheres 4](#_Toc340101844)

[LO: Loading multiple model data using a loader 5](#_Toc340101845)

[LO: Moving models in a scene 8](#_Toc340101846)

### 

### LO: Creating a skybox in Three.js

**The goal:**

To create a realistic and friendly skybox that surrounds the game area to make the surroundings believable.

**Problem:**

* Not knowing how to load in textures or images.
* Not knowing how to apply the texture to a rendering primitive so it can be displayed in the scene.
* Not knowing the co-ordinate system to place the primitive.

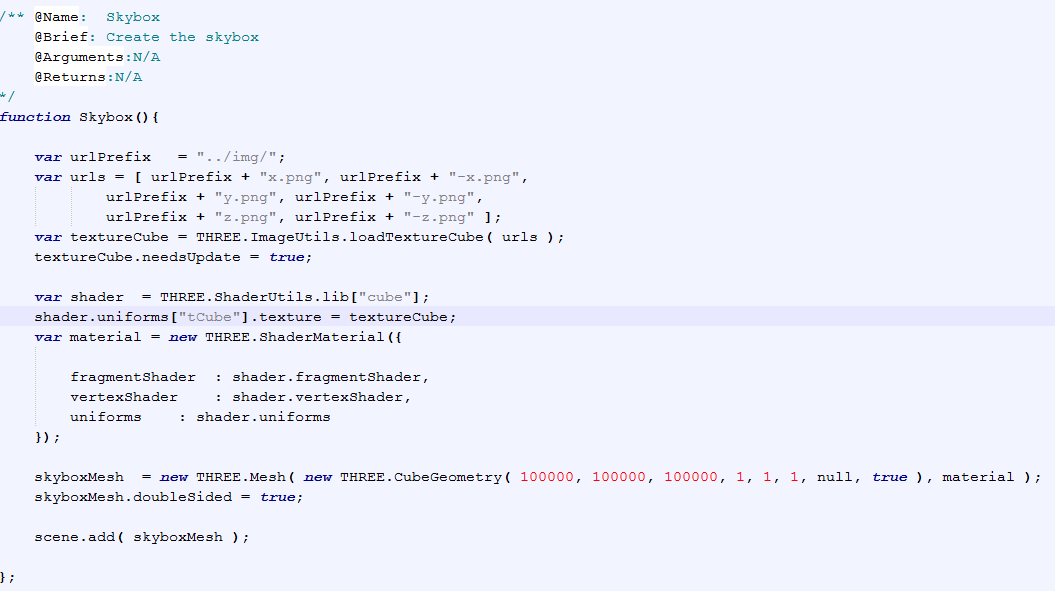
**Solution:**

I researched in many places to figure out how to go about doing this. Some of the more useful resource links were:

* <http://stemkoski.github.com/Three.js/>
* <http://learningthreejs.com/data/lets_do_a_sky/docs/lets_do_a_sky.html>
* <http://stackoverflow.com/questions/12670183/three-js-skybox-texture-issue>
* <http://mrdoob.github.com/three.js/>
* <http://www.zfight.com/misc/images/textures/envmaps/stormydays_large.jpg>

Here is the code snippet I came up with. It involved editing the texture in the previous link above into individual sides of the sky cube and applying them to their own square.

Then it was a matter of placing them in the scene to be rendered.



### LO: Adding models to the scene

**The goal:**

To add meshes to the scene that can be used to represent objects, be they characters or static objects.

**Problem:**

* Not knowing how to create a model.
* Not knowing how to define its size.
* Not knowing the required files needed.

**Solution:**

I researched in many places to figure out how to go about doing this. Some of the more useful resource links were:

* Making a mesh <http://www.aerotwist.com/tutorials/getting-started-with-three-js/>
* Shaped objects <http://mrdoob.github.com/three.js/examples/webgl_interactive_cubes.html>

Here is the code snippet I came up with. In this instance it is a tree billboard. I load an image of a tree and apply it to a plane geometry I created which defines the size of the physical tree in game.

You will notice that I used this same geometry and created several meshes. One created I can position and rotate the mesh any way I choose.

Then it was a matter of placing them in the scene to be rendered.



### LO: Collision detection using Three.js bounding spheres

**The goal:**

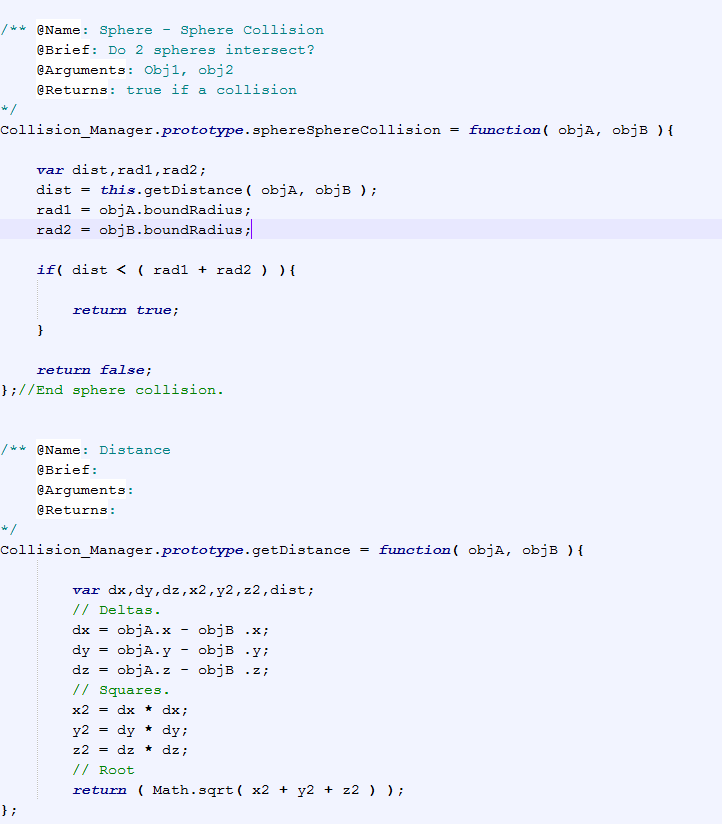
To implement some form of basic collision detection so that the player has limitations of where they can move and so they could pick up items in game. The goal here is to keep it as simple as possible at first.

**Problem:**

* Figuring out how to get the position of a mesh and access its bounding sphere if it had one.

**Solution:**

Using the debugger in chrome I inspected each mesh object created so far to try and see if I could find something I could use before creating a bounding sphere for each object. Low and behold there it was; bound radius! I then used some simple calculations to see if the distance between the objects was less than the combined radii. If it was then a collision occurred.



### LO: Loading multiple model data using a loader

**The goal:**

To load collada models exported from blender into the game efficiently.

**Problem:**

As it stood I could load in the models but because the Collada loader packaged with Three.js used Ajax to read from file; the callbacks interrupted each other and I lost scope of the ’this’ pointer in JS.

When the callbacks returned there was no way to tell what model it was as I couldn’t access a name or it was blank and not even the url of the file loaded could be seen.

**Solution:**

To combat that I created a mesh loader that will load the game models sequentially and take their name before loading and applying it when the callback is fired. Then move on to the nest model in the load cache etc…

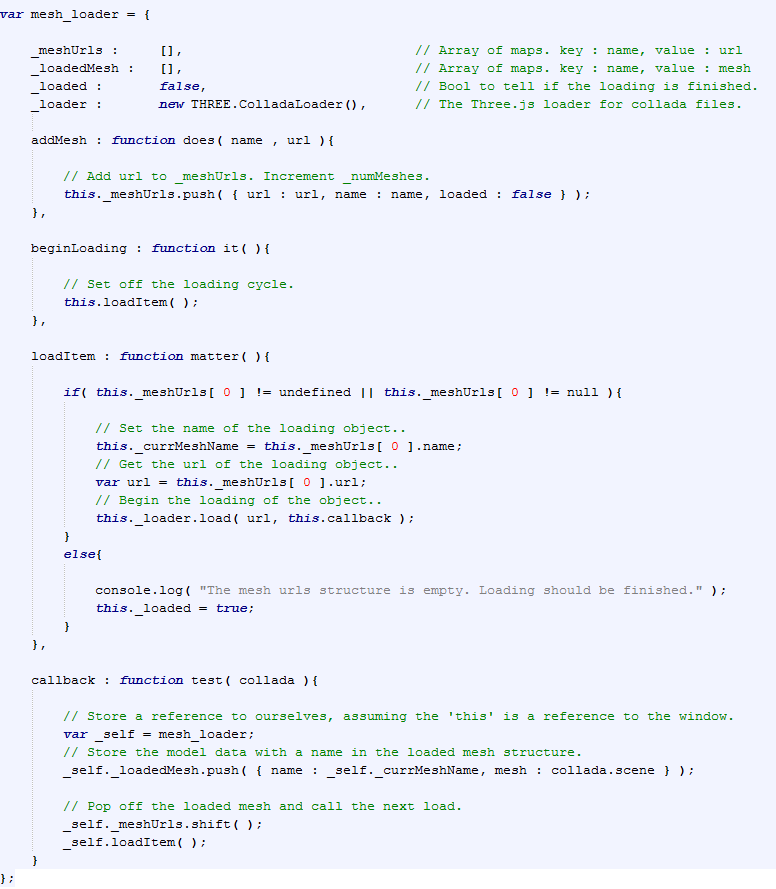


Figure The loader class itself

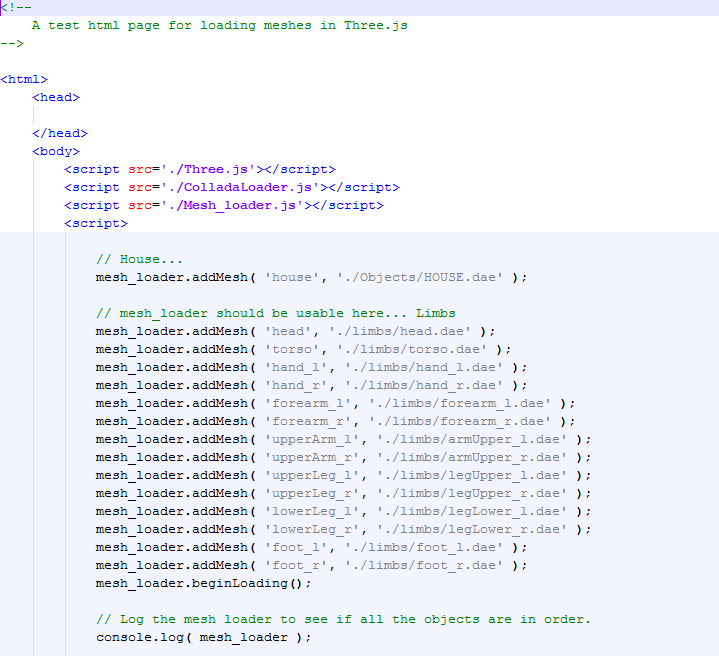


Figure The corresponding html usage

### LO: Moving models in a scene

**The goal:**

After loading a model in the scene I had to find out how to move it as not all the meshes are static.

An example of this is the limbs for the player’s body. They have to follow the position of the player at all times as they “are” the player.

**Problem:**

* Not knowing how to move a model.
* Not sure what type of model I had to create to move it.

**Solution:**

Some of the links I researched this problem:

* <http://catchvar.com/threejs-animating-blender-models>
* <http://www.96methods.com/2012/01/three-js-moving-objects/>
* Moving the camera object <http://www.96methods.com/2011/12/three-js-moving-the-camera/>
* And a more simple way <http://stackoverflow.com/questions/11127543/how-to-move-an-object-forward-in-three-js>

The code I went with moves the limb models according to the Kinect data got from the user in real life. As they are vectors I can apply them after a translation and that’s what I did.

Here is the sample snippet:

